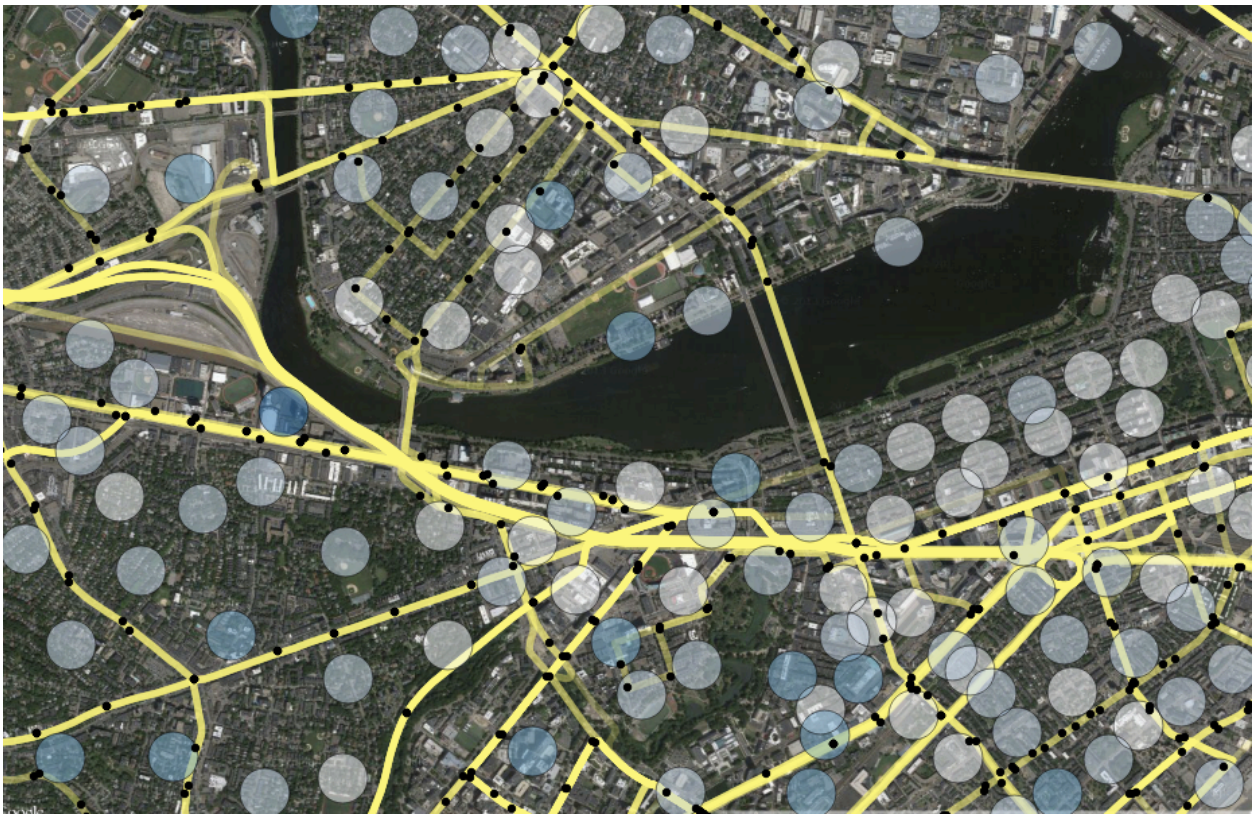
THE WORLD BANK
Working for a World Free of Poverty

Open Source Tools For Transport Planning: Indicator Calculation Methodology Report



Prepared by

azavea
340 N 12th St, Suite 402
Philadelphia, PA 19107
(215) 925-2600
http://www.azavea.com

Table of Contents

The World Bank

Open Source Tools for Transport Planning

Indicator Calculation Methodology

The calculation methods outlined in this document provide examples of how to measure indicators using a spatially enabled SQL database. These examples do not necessarily prescribe the exact approach that full-fledged software will take, but provide a proof-of-concept and logical structure for calculations. With more robust software, there will likely be performance enhancements related to memory caching and multithreading that can be taken advantage of.

Number of modes and types

Required Data: GTFS
Tables Accessed: Routes, Route types
Discussion: The route types table contains friendly names of the types of transit used in the system. This query joins the description to the routes table, counts the number of individual routes, and groups by the route type.

SQL:

```
SELECT t.description, COUNT(r.*) as numlines FROM gtfs_routes r
JOIN gtfs_route_types t ON r.route_type = t.route_type
GROUP BY t.description;
```

Transit system length in km

Required Data: GTFS
Tables Accessed: Shapes or Stop times
Discussion: The shapes.txt GTFS file is optional, as is the stop_times shape_distance_traveled field. This indicator will not be available for GTFS data that includes neither shapes nor stop distance traveled.

This series of queries creates and spatially enables two new tables in the database. The shapelines table contains a line for each shape_id from the shapes.txt table. Subsequent queries calculate line length and group by route, mode, and system.

SQL:

```
--This creates a polyline table with the local UTM projection
CREATE TABLE shapelines (shape_id VARCHAR, len_ft NUMERIC);
SELECT AddGeometryColumn ( 'shapelines', 'geom', 32619, 'LINESTRING', 2);

--Add a local projection column to the points
SELECT AddGeometryColumn ( 'gtfs_shapes', 'geom', 32619, 'POINT', 2);
```

```
UPDATE gtfs_shapes SET geom = ST_Transform(ST_SetSRID(ST_MakePoint(shape_pt_lon,
shape_pt_lat),4326),32619);

--Builds the lines
INSERT INTO shapelines (SELECT shape_id, 0.0, ST_MakeLine(ST_SnapToGrid(geom, 1.0)
ORDER BY shape_pt_sequence) as newgeom FROM gtfs_shapes GROUP BY shape_id);
UPDATE shapelines SET len_meters = ST_Length(geom);



-- length by mode if there's a shapes.txt
SELECT description, SUM(avg_len) as modelength
FROM
(SELECT r.route_id, AVG(s.len_meters/1000) as avg_len, r.route_type
FROM shapelines s
JOIN gtfs_trips t ON s.shape_id = t.shape_id
JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY r.route_id, s.len_meters, r.route_type
) as lengths
JOIN gtfs_route_types USING(route_type)
GROUP BY description

-- if no shapes.txt and stop_distance_traveled is populated
SELECT description, SUM(avg_len) as modelength
FROM
(SELECT r.route_id, AVG(s.maxstopdist) as avg_len, r.route_type
FROM (SELECT trip_id, MAX(shape_dist_traveled)/1000 AS maxstopdist
FROM gtfs_stop_times GROUP BY trip_id) s
JOIN gtfs_trips t ON t.trip_id = s.trip_id
JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY r.route_id, r.route_type
) as lengths
JOIN gtfs_route_types USING(route_type)
GROUP BY description
```

Number of stops

Required Data: GTFS

Tables Accessed: Routes, trips, and stop times

Discussion: This query counts the number of stops per route or mode.

SQL:

```
--By Route
```

```
SELECT route_id, MAX(number) as num_stops
FROM
(SELECT t.trip_id, r.route_id, COUNT(*) as number
FROM gtfs_stop_times st
LEFT JOIN gtfs_trips t ON st.trip_id = t.trip_id
LEFT JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY t.trip_id, r.route_id
) as route_stops
GROUP BY route_id;

--By Mode
SELECT gtfs_route_types.description, COUNT(*)
FROM
(SELECT st.stop_id, r.route_type, COUNT(*) as number
FROM gtfs_stop_times st
LEFT JOIN gtfs_trips t ON st.trip_id = t.trip_id
LEFT JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY st.stop_id, r.route_type
) as route_stops
JOIN gtfs_route_types USING(route_type)
GROUP BY gtfs_route_types.description;
```

Weekly number of hours of service

Required Data: GTFS
Tables Accessed: Routes, trips, stop times, calendar, calendar dates, frequencies
Discussion: The GTFS definition recommends that stop times after midnight not wrap around the day change (going from 23:59 to 00:00), but instead continue above the 24-hour mark. This calculation depends on the data following this recommendation.

The SQL query below only gets hours of service for Monday. The query will be repeated for each day of the week, and the duration for each day can be summed to get the weekly number or hours of service. The query controls for some routes being operated on a frequency basis.
SQL:

```
--For individual routes
-- Duration of Service (for a single weekday)
-- By route
SELECT mins.route_id, maxs.maxtime - mins.mintime as duration FROM
(SELECT MIN(mintime) as mintime, route_id, route_type FROM

(SELECT MIN(CAST(SUBSTRING(arrival_time FROM 0 FOR 3) AS INTEGER)) as mintime,
r.route_id, r.route_type
FROM gtfs_stop_times st
```

```
JOIN gtfs_trips t ON st.trip_id = t.trip_id
JOIN gtfs_routes r on r.route_id = t.route_id
WHERE t.service_id IN (SELECT service_id FROM gtfs_calendar WHERE monday = 1) AND NOT
EXISTS (SELECT gtfs_frequencies.trip_id FROM gtfs_frequencies WHERE st.trip_id =
gtfs_frequencies.trip_id)
GROUP BY r.route_id, r.route_type

UNION
SELECT MIN(CAST(SUBSTRING(start_time FROM 0 FOR 3) AS INTEGER)) as mintime, r.route_id,
r.route_type
FROM gtfs_frequencies f
JOIN gtfs_trips t ON f.trip_id = t.trip_id
JOIN gtfs_routes r on r.route_id = t.route_id
WHERE t.service_id IN (SELECT service_id FROM gtfs_calendar WHERE monday = 1)
GROUP BY r.route_id, r.route_type
) as freq_stop_mins GROUP BY route_id, route_type) as mins
JOIN
(SELECT MAX(maxtime) as maxtime, route_id, route_type FROM

(SELECT MAX(CAST(SUBSTRING(arrival_time FROM 0 FOR 3) AS INTEGER)) as maxtime,
r.route_id, r.route_type
FROM gtfs_stop_times st
JOIN gtfs_trips t ON st.trip_id = t.trip_id
JOIN gtfs_routes r on r.route_id = t.route_id
WHERE t.service_id IN (SELECT service_id FROM gtfs_calendar WHERE monday = 1) AND NOT
EXISTS (SELECT gtfs_frequencies.trip_id FROM gtfs_frequencies WHERE st.trip_id =
gtfs_frequencies.trip_id)
GROUP BY r.route_id, r.route_type

UNION
SELECT MAX(CAST(SUBSTRING(end_time FROM 0 FOR 3) AS INTEGER)) as maxtime, r.route_id,
r.route_type
FROM gtfs_frequencies f
JOIN gtfs_trips t ON f.trip_id = t.trip_id
JOIN gtfs_routes r on r.route_id = t.route_id
WHERE t.service_id IN (SELECT service_id FROM gtfs_calendar WHERE monday = 1)
GROUP BY r.route_id, r.route_type
) as freq_stop_maxs GROUP BY route_id, route_type) as maxs
ON mins.route_id = maxs.route_id;

-- By mode
SELECT route_types.description, MAX(maxs.maxtime) - MIN(mins.mintime) as duration
FROM
```

```
(SELECT mintime, route_id, route_type FROM
(SELECT MIN(CAST(SUBSTRING(arrival_time FROM 0 FOR 3) AS INTEGER)) as mintime,
r.route_id, r.route_type
              FROM gtfs_stop_times st
              JOIN gtfs_trips t ON st.trip_id = t.trip_id
              JOIN gtfs_routes r on r.route_id = t.route_id
              WHERE t.service_id IN (SELECT service_id FROM gtfs_calendar WHERE monday
= 1) AND NOT EXISTS (SELECT gtfs_frequencies.trip_id FROM gtfs_frequencies WHERE
st.trip_id = gtfs_frequencies.trip_id)
GROUP BY r.route_id, r.route_type
UNION
SELECT MIN(CAST(SUBSTRING(start_time FROM 0 FOR 3) AS INTEGER)) as mintime, r.route_id,
r.route_type
FROM gtfs_frequencies f
JOIN gtfs_trips t ON f.trip_id = t.trip_id
JOIN gtfs_routes r ON r.route_id = t.route_id
WHERE t.service_id IN (SELECT service_id FROM gtfs_calendar WHERE monday = 1)
GROUP BY r.route_id, r.route_type
) AS freq_stop_mins GROUP BY route_id, route_type, mintime) as mins

JOIN

(SELECT maxtime, route_id, route_type FROM

       (SELECT MAX(CAST(SUBSTRING(arrival_time FROM 0 FOR 3) AS INTEGER)) as maxtime,
r.route_id, r.route_type
              FROM gtfs_stop_times st
              JOIN gtfs_trips t ON st.trip_id = t.trip_id
              JOIN gtfs_routes r on r.route_id = t.route_id
              WHERE t.service_id IN (SELECT service_id FROM gtfs_calendar WHERE monday
= 1) AND NOT EXISTS (SELECT gtfs_frequencies.trip_id FROM gtfs_frequencies WHERE
st.trip_id = gtfs_frequencies.trip_id)
GROUP BY r.route_id, r.route_type
UNION
SELECT MAX(CAST(SUBSTRING(end_time FROM 0 FOR 3) AS INTEGER)) as maxtime, r.route_id,
r.route_type
FROM gtfs_frequencies f
JOIN gtfs_trips t ON f.trip_id = t.trip_id
JOIN gtfs_routes r ON r.route_id = t.route_id
WHERE t.service_id IN (SELECT service_id FROM gtfs_calendar WHERE monday = 1)
GROUP BY r.route_id, r.route_type
) AS freq_stop_maxs GROUP BY route_id, route_type, maxtime) as maxs
ON mins.route_id = maxs.route_id
```

```
JOIN

(SELECT * FROM gtfs_route_types) as route_types ON mins.route_type =
route_types.route_type
GROUP BY route_types.description;
```

Ratio of number of stops to route-length

Required Data: GTFS

Tables Accessed: Routes, trips, shapes or stop times

Discussion: There are two ways to derive the length of a route, as shown above. The below queries rely on either a shapes.txt file or the stop_distance_traveled field in the stops_times.txt field.

SQL:

```
SELECT lenavgs.route_id, stopscount.num_stops / lenavgs.routelength AS coverage
FROM
(SELECT route_id, route_type, AVG(len_meters/1000) as routelength
FROM
(SELECT s.shape_id, s.len_meters, t.route_id, r.route_type
FROM shapelines s
JOIN gtfs_trips t ON s.shape_id = t.shape_id
JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY s.shape_id, s.len_meters, t.route_id, r.route_type
) as lengths
GROUP BY route_id, route_type) as lenavgs
JOIN
(SELECT route_id, MAX(number) as num_stops
FROM
(SELECT t.trip_id, r.route_id, COUNT(*) as number
FROM gtfs_stop_times st
LEFT JOIN gtfs_trips t ON st.trip_id = t.trip_id
LEFT JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY t.trip_id, r.route_id
) as route_stops
GROUP BY route_id) as stopscount
ON lenavgs.route_id = stopscount.route_id;


--by mode
SELECT gtfs_route_types.description, stopssum.numbstops / lenavgs.modelength AS
coverage
FROM
(SELECT route_type, SUM(avg_len) as modelength
```

```
FROM
(SELECT r.route_id, AVG(s.len_meters/1000) as avg_len, r.route_type
FROM shapelines s
JOIN gtfs_trips t ON s.shape_id = t.shape_id
JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY r.route_id, s.len_meters, r.route_type
) as lengths
GROUP BY route_type) as lenavgs
JOIN
(SELECT route_type, SUM(num_stops) as numbstops
FROM
(SELECT route_id, route_type, MAX(number) as num_stops
FROM
(SELECT t.trip_id, r.route_id, route_type, COUNT(*) as number
FROM gtfs_stop_times st
LEFT JOIN gtfs_trips t ON st.trip_id = t.trip_id
LEFT JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY t.trip_id, r.route_id, route_type
) as route_stops
GROUP BY route_type, route_id) as stopscount
GROUP BY route_type) as stopssum
ON lenavgs.route_type = stopssum.route_type
JOIN gtfs_route_types ON lenavgs.route_type = gtfs_route_types.route_type;
--for when there's no shapes.txt
SELECT lenavgs.route_id, stopscount.num_stops / lenavgs.routelength AS coverage
FROM
(SELECT r.route_id, AVG(s.maxstopdist) as routelength, r.route_type
FROM (SELECT trip_id, MAX(shape_dist_traveled)/1000 AS maxstopdist
FROM gtfs_stop_times GROUP BY trip_id) s
JOIN gtfs_trips t ON t.trip_id = s.trip_id
JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY r.route_id, r.route_type
) as lenavgs
JOIN
(SELECT route_id, MAX(number) as num_stops
FROM
(SELECT t.trip_id, r.route_id, COUNT(*) as number
FROM gtfs_stop_times st
LEFT JOIN gtfs_trips t ON st.trip_id = t.trip_id
LEFT JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY t.trip_id, r.route_id
) as route_stops
GROUP BY route_id) as stopscount
```

```
ON lenavgs.route_id = stopscount.route_id;

--by mode
SELECT gtfs_route_types.description, stopssum.numbstops / lenavgs.modelength AS
coverage
FROM
(SELECT route_type, SUM(routelength) as modelength
FROM
(SELECT r.route_id, AVG(s.maxstopdist) as routelength, r.route_type
FROM (SELECT trip_id, MAX(shape_dist_traveled)/1000 AS maxstopdist
FROM gtfs_stop_times GROUP BY trip_id) s
JOIN gtfs_trips t ON t.trip_id = s.trip_id
JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY r.route_id, r.route_type
) as lengths
GROUP BY route_type) as lenavgs
JOIN
(SELECT route_type, SUM(num_stops) as numbstops
FROM
(SELECT route_id, route_type, MAX(number) as num_stops
FROM
(SELECT t.trip_id, r.route_id, route_type, COUNT(*) as number
FROM gtfs_stop_times st
LEFT JOIN gtfs_trips t ON st.trip_id = t.trip_id
LEFT JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY t.trip_id, r.route_id, route_type
) as route_stops
GROUP BY route_type, route_id) as stopscount
GROUP BY route_type) as stopssum
ON lenavgs.route_type = stopssum.route_type
JOIN gtfs_route_types ON lenavgs.route_type = gtfs_route_types.route_type;
```

Average distance between stops

Required Data: GTFS

Tables Accessed: Routes, trips, shapes or stop times

Discussion: Similar to the above indicator, there are two ways to calculate this indicator, depending on data available.

SQL:

```
-- works if shape_dist_traveled is populated
-- By Route
SELECT lenavgs.route_id, lenavgs.routelength / stopscount.num_stops AS coverage
FROM
```

```
(SELECT r.route_id, AVG(s.maxstopdist) as routelength, r.route_type
FROM (SELECT trip_id, MAX(shape_dist_traveled)/1000 AS maxstopdist
FROM gtfs_stop_times GROUP BY trip_id) s
JOIN gtfs_trips t ON t.trip_id = s.trip_id
JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY r.route_id, r.route_type
) as lenavgs
JOIN
(SELECT route_id, MAX(number) as num_stops
FROM
(SELECT t.trip_id, r.route_id, COUNT(*) as number
FROM gtfs_stop_times st
LEFT JOIN gtfs_trips t ON st.trip_id = t.trip_id
LEFT JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY t.trip_id, r.route_id
) as route_stops
GROUP BY route_id) as stopscount
ON lenavgs.route_id = stopscount.route_id;

-- By Mode
SELECT gtfs_route_types.description, lenavgs.modelength / stopssum.numbstops AS
coverage
FROM
(SELECT route_type, SUM(routelength) as modelength
FROM
(SELECT r.route_id, AVG(s.maxstopdist) as routelength, r.route_type
FROM (SELECT trip_id, MAX(shape_dist_traveled)/1000 AS maxstopdist
FROM gtfs_stop_times GROUP BY trip_id) s
JOIN gtfs_trips t ON t.trip_id = s.trip_id
JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY r.route_id, r.route_type
) as lengths
GROUP BY route_type) as lenavgs
JOIN
(SELECT route_type, SUM(num_stops) as numbstops
FROM
(SELECT route_id, route_type, MAX(number) as num_stops
FROM
(SELECT t.trip_id, r.route_id, route_type, COUNT(*) as number
FROM gtfs_stop_times st
LEFT JOIN gtfs_trips t ON st.trip_id = t.trip_id
LEFT JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY t.trip_id, r.route_id, route_type
```

```
) as route_stops
GROUP BY route_type, route_id) as stopscount
GROUP BY route_type) as stopssum
ON lenavgs.route_type = stopssum.route_type
JOIN gtfs_route_types ON lenavgs.route_type = gtfs_route_types.route_type;

--otherwise, need to use shape length
-- By Route
SELECT lenavgs.route_id, lenavgs.routelength / stopscount.num_stops AS coverage
FROM
(SELECT route_id, route_type, AVG(len_meters/1000) as routelength
FROM
(SELECT s.shape_id, s.len_meters, t.route_id, r.route_type
FROM shapelines s
JOIN gtfs_trips t ON s.shape_id = t.shape_id
JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY s.shape_id, s.len_meters, t.route_id, r.route_type
) as lengths
GROUP BY route_id, route_type) as lenavgs
JOIN
(SELECT route_id, MAX(number) as num_stops
FROM
(SELECT t.trip_id, r.route_id, COUNT(*) as number
FROM gtfs_stop_times st
LEFT JOIN gtfs_trips t ON st.trip_id = t.trip_id
LEFT JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY t.trip_id, r.route_id
) as route_stops
GROUP BY route_id) as stopscount
ON lenavgs.route_id = stopscount.route_id;

-- By Mode
SELECT gtfs_route_types.description, lenavgs.modelength / stopssum.numbstops AS
coverage
FROM
(SELECT route_type, SUM(avg_len) as modelength
FROM
(SELECT r.route_id, AVG(s.len_meters/1000) as avg_len, r.route_type
FROM shapelines s
JOIN gtfs_trips t ON s.shape_id = t.shape_id
JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY r.route_id, s.len_meters, r.route_type
) as lengths
```

```
GROUP BY route_type) as lenavgs
JOIN
(SELECT route_type, SUM(num_stops) as numbstops
FROM
(SELECT route_id, route_type, MAX(number) as num_stops
FROM
(SELECT t.trip_id, r.route_id, route_type, COUNT(*) as number
FROM gtfs_stop_times st
LEFT JOIN gtfs_trips t ON st.trip_id = t.trip_id
LEFT JOIN gtfs_routes r on r.route_id = t.route_id
GROUP BY t.trip_id, r.route_id, route_type
) as route_stops
GROUP BY route_type, route_id) as stopscount
GROUP BY route_type) as stopssum
ON lenavgs.route_type = stopssum.route_type
JOIN gtfs_route_types ON lenavgs.route_type = gtfs_route_types.route_type;
```

Average weekday peak, non-peak, and aggregated time traveled between stops

Required Data: GTFS

Tables Accessed: Routes, trips, stop times, calendar

Discussion: This indicator averages the time it takes to get between consecutive stops along a route, aggregated at different levels. Including the day of the week and limits on stop times narrow the focus to particular blocks of time. The below query is the average without regard to time.

SQL:

```
-- by route
SELECT d.route_id, AVG((b.maxtime - a.mintime) * 60 / c.numstops) AS
avg_min_between_stops
FROM (SELECT trip_id, MIN(CAST(substring(arrival_time from 1 for 2) as decimal) +
cast(substring(arrival_time from 4 for 2) as decimal)/60) as mintime,
min(stop_sequence) as minseq FROM gtfs_stop_times GROUP BY trip_id ORDER BY trip_id) a
JOIN (SELECT trip_id, MAX(CAST(substring(arrival_time from 1 for 2) as decimal) +
cast(substring(arrival_time from 4 for 2) as decimal)/60) as maxtime,
max(stop_sequence) as maxseq FROM gtfs_stop_times GROUP BY trip_id ORDER BY trip_id) b
ON a.trip_id = b.trip_id
JOIN (SELECT trip_id, COUNT(*) as numstops FROM gtfs_stop_times GROUP BY trip_id ORDER
BY trip_id) c ON a.trip_id = c.trip_id
JOIN gtfs_trips d ON d.trip_id = a.trip_id
GROUP BY d.route_id

-- by mode
```

```
SELECT f.description, AVG((b.maxtime - a.mintime) * 60 / c.numstops) AS
avg_min_between_stops
FROM (SELECT trip_id, MIN(CAST(substring(arrival_time from 1 for 2) as decimal) +
cast(substring(arrival_time from 4 for 2) as decimal)/60) as mintime,
min(stop_sequence) as minseq FROM gtfs_stop_times GROUP BY trip_id ORDER BY trip_id) a
JOIN (SELECT trip_id, MAX(CAST(substring(arrival_time from 1 for 2) as decimal) +
cast(substring(arrival_time from 4 for 2) as decimal)/60) as maxtime,
max(stop_sequence) as maxseq FROM gtfs_stop_times GROUP BY trip_id ORDER BY trip_id) b
ON a.trip_id = b.trip_id
JOIN (SELECT trip_id, COUNT(*) as numstops FROM gtfs_stop_times GROUP BY trip_id ORDER
BY trip_id) c ON a.trip_id = c.trip_id
JOIN gtfs_trips d ON d.trip_id = a.trip_id
JOIN gtfs_routes e ON d.route_id = e.route_id
JOIN gtfs_route_types f ON e.route_type = f.route_type
GROUP BY f.description
```

Average system weekday / weekend frequency

Required Data: GTFS

Tables Accessed: Routes, trips, stop times, calendar, frequencies

Discussion: Compiles the headway for a specific stop and route combination, then averages by route and mode. Weekend and weekday options can be selected by modifying the join to the calendar table. This query accommodates GTFS data that relies on frequency-based scheduling using the frequencies.txt table.

SQL:

```
-- Average service frequency, by route.
-- The actual calculation is (1/(sum/duration))*60*60 to get from headway in seconds to
frequency in hours
-- Formally, the headway is the period, defined as the reciprocal of the frequency.
SELECT 3600*(srv_end-srv_start)/(sum) as buses_per_hr, route_id FROM

(SELECT SUM(trip_headway_hrs_wghted), MIN(starttime) as srv_start, MAX(endtime) as
srv_end, route_id FROM
(SELECT (endtime-starttime)*headway_secs as trip_headway_hrs_wghted, starttime,
endtime, trip_id, route_id FROM
(SELECT start_time, end_time, freqs.trip_id, trips.route_id, headway_secs,
       CAST(substring(start_time FROM 1 FOR 2) as decimal) +
       CAST(substring(start_time FROM 4 for 2) as decimal)/60 AS starttime,
       CAST(substring(end_time FROM 1 FOR 2) as decimal) +
       CAST(substring(end_time FROM 4 for 2) as decimal)/60 AS endtime
       FROM gtfs_frequencies as freqs

JOIN
```

```
(SELECT trip_id, service_id, route_id FROM gtfs_trips) AS trips ON
trips.trip_id=freqs.trip_id

JOIN

(SELECT service_id, monday, tuesday, wednesday, thursday, friday FROM gtfs_calendar
       WHERE monday='1' OR tuesday='1' OR wednesday='1' OR thursday='1' OR friday='1'
) AS weekdays
ON weekdays.service_id=trips.service_id) AS trip_route_start_end_hrs
) AS wghted_headways
GROUP BY route_id) as agg_components

UNION --Connect stop-based counting to frequency-based counting

select avg(num/duration_hrs) as buses_per_hr, rt_stp_duration.route_id from
--Get the duration (and some other stats) separately
(select (max(rt_stp_hrs.arrtime) - min(rt_stp_hrs.arrtime)) as duration_hrs, count(*)
as num, rt_stp_hrs.route_id, rt_stp_hrs.stop_id from
       --Gives all arrivals for one stop without regard to day.
(select distinct gtfs_stop_times.trip_id, arrival_time, departure_time, stop_id,
trips.route_id,
           CAST(substring(arrival_time from 1 for 2) as decimal) +
           cast(substring(arrival_time from 4 for 2) as decimal)/60 as arrtime
       from gtfs_stop_times
       --Gives the trip_id to join the next query on
join(select trip_id, service_id, route_id from gtfs_trips) as trips on
trips.trip_id=gtfs_stop_times.trip_id
--Gives all trips which occur on the specified days.
join(select service_id, monday, tuesday, wednesday, thursday, friday
           from gtfs_calendar
           where monday='1' or tuesday='1' or wednesday='1' or thursday='1' or
friday='1') as weekdays
on weekdays.service_id=trips.service_id
--Filter out anything that's in frequencies
WHERE NOT EXISTS (SELECT gtfs_frequencies.trip_id FROM gtfs_frequencies WHERE
gtfs_stop_times.trip_id = gtfs_frequencies.trip_id)
order by arrtime, stop_id) as rt_stp_hrs group by rt_stp_hrs.route_id,
rt_stp_hrs.stop_id
) as rt_stp_duration where rt_stp_duration.duration_hrs > 0.05 group by
rt_stp_duration.route_id
;
```

```
--by mode
SELECT AVG(buses_per_hr), description FROM

((SELECT 3600*(srv_end-srv_start)/(sum) as buses_per_hr, route_id FROM

(SELECT SUM(trip_headway_hrs_wghted), MIN(starttime) as srv_start, MAX(endtime) as
srv_end, route_id FROM
(SELECT (endtime-starttime)*headway_secs as trip_headway_hrs_wghted, starttime,
endtime, trip_id, route_id FROM
(SELECT start_time, end_time, freqs.trip_id, trips.route_id, headway_secs,
       CAST(substring(start_time FROM 1 FOR 2) as decimal) +
       CAST(substring(start_time FROM 4 for 2) as decimal)/60 AS starttime,
       CAST(substring(end_time FROM 1 FOR 2) as decimal) +
       CAST(substring(end_time FROM 4 for 2) as decimal)/60 AS endtime
       FROM gtfs_frequencies as freqs

JOIN

(SELECT trip_id, service_id, route_id FROM gtfs_trips) AS trips ON
trips.trip_id=freqs.trip_id

JOIN

(SELECT service_id, monday, tuesday, wednesday, thursday, friday FROM gtfs_calendar
       WHERE monday='1' OR tuesday='1' OR wednesday='1' OR thursday='1' OR friday='1'
) AS weekdays
ON weekdays.service_id=trips.service_id) AS trip_route_start_end_hrs
) AS wghted_headways
GROUP BY route_id) as agg_components

UNION --Connect stop-based counting to frequency-based counting

select avg(num/duration_hrs) as buses_per_hr, rt_stp_duration.route_id from
--Get the duration (and some other stats) separately
(select (max(rt_stp_hrs.arrtime) - min(rt_stp_hrs.arrtime)) as duration_hrs, count(*)
as num, rt_stp_hrs.route_id, rt_stp_hrs.stop_id from
       --Gives all arrivals for one stop without regard to day.
(select distinct gtfs_stop_times.trip_id, arrival_time, departure_time, stop_id,
trips.route_id,
           CAST(substring(arrival_time from 1 for 2) as decimal) +
           cast(substring(arrival_time from 4 for 2) as decimal)/60 as arrtime
       from gtfs_stop_times
       --Gives the trip_id to join the next query on
```

```
join(select trip_id, service_id, route_id from gtfs_trips) as trips on
trips.trip_id=gtfs_stop_times.trip_id
--Gives all trips which occur on the specified days.
join(select service_id, monday, tuesday, wednesday, thursday, friday
           from gtfs_calendar
           where monday='1' or tuesday='1' or wednesday='1' or thursday='1' or
friday='1') as weekdays
on weekdays.service_id=trips.service_id
--Filter out anything that's in frequencies
WHERE NOT EXISTS (SELECT gtfs_frequencies.trip_id FROM gtfs_frequencies WHERE
gtfs_stop_times.trip_id = gtfs_frequencies.trip_id)
order by arrtime, stop_id) as rt_stp_hrs group by rt_stp_hrs.route_id,
rt_stp_hrs.stop_id
) as rt_stp_duration where rt_stp_duration.duration_hrs > 0.05 group by
rt_stp_duration.route_id
) as avg_freq_by_rt

JOIN

(SELECT route_id, route_type FROM gtfs_routes) AS types ON types.route_id =
avg_freq_by_rt.route_id

JOIN

(SELECT route_type, description FROM gtfs_route_types) as descs ON
types.route_type=descs.route_type
) AS freq_with_descs
GROUP BY description;
```

System Affordability as a Proportion of Poverty Line Income

Required Data: GTFS, road network

Tables Accessed: Fare attributes, fare rules

Discussion: Where fare information is not included in the GTFS data, an average one-way fare can be during software configuration. The income threshold defining poverty will also be collected during configuration. The threshold in the United States for an individual is $11,490

```
-- Rides per month = 42, Fare = $2.50, Poverty line = $11490 divided by 12 months
SELECT (42 * 2.5) / (11490 / 12) * 100;
```

Ratio of Total Transit Lines Length over Total Road Network Length (RBR)

Required Data: GTFS, road network, regional boundary
Tables Accessed: Shapes, roads, region
Discussion: This set of queries buffers the transit system shape by 3m, then clips the road network that falls within that buffer. This is the measure of roadway covered by the transit system, which is divided by the total road network length, each relative to the region or city boundary. In the software, this sequence will need significant optimization, as the SQL performance is quite slow.
SQL:

```
-- First calculate the length of road segments
ALTER TABLE roads ADD COLUMN len_meters NUMERIC;
UPDATE roads SET len_meters = ST_Length(geom);

-- Create a spatial index on shapelines
CREATE INDEX spidx_shapelines ON shapelines USING GIST(geom);

-- Create a spatial index on roads
CREATE INDEX spidx_roads ON roads USING GIST(geom);


-- Takes a long time
CREATE TABLE shapesbuffer AS SELECT ST_Union(ST_Buffer(shapelines.geom,3)) as geom FROM
shapelines;

SELECT DISTINCT
(SELECT SUM(len_meters/1000)
FROM
       (SELECT ST_Intersection(shapesbufferunion.geom,roads.geom), len_meters
       FROM shapesbufferunion, roads
       WHERE ST_Intersects(shapesbufferunion.geom,roads.geom)) as clip) /
(SELECT SUM(len_meters/1000)
FROM
       (SELECT ST_Intersection(region.geom,roads.geom), len_meters
       FROM region, roads
       WHERE ST_Intersects(region.geom,roads.geom)) as clip)
FROM shapelines;
```

Transit Line Network Density (BND)

Required Data: Shapes, regional boundary
Tables Accessed: Shapes, region

Discussion: The result of this indicator is an index representing kilometers of transit lines over square kilometers of area. The query selects a subset of lines where they intersect the regional and city boundaries. The line lengths are calculated in the same way as the above query.
SQL:

```
-- for county boundary
SELECT (SELECT SUM(len_meters)/1000 FROM
(SELECT ST_Intersection(shapesbufferunion.geom,roads.geom) as geom, len_meters
       FROM shapesbufferunion, roads
       WHERE ST_Intersects(shapesbufferunion.geom,roads.geom)) as clip, region
WHERE ST_Intersects(region.geom,clip.geom)) / (SELECT SUM(ST_Area(geom))/1000000 from
region) as BND;

-- for city boundary
SELECT (SELECT SUM(len_meters)/1000 FROM
(SELECT ST_Intersection(shapesbufferunion.geom,roads.geom) as geom, len_meters
       FROM shapesbufferunion, roads
       WHERE ST_Intersects(shapesbufferunion.geom,roads.geom)) as clip, city
WHERE ST_Intersects(city.geom,shapelines.geom)) / (SELECT SUM(ST_Area(geom))/1000000
from city) as BND;
```

Coverage of Transit Stops for 500 meter Radius (CBS_500)

Required Data: Stops, boundary files
Tables Accessed: Stops, region, city
Discussion: The stops table must first be given a spatial column, if it does not already have one. The calculation unions the 500m buffers of each stop, then divides the area by the total regional or city area. Only stops within the comparison area are buffered. If the system has lines using informal stops, a 500m buffer of the lines themselves can be used.
SQL:

```
SELECT AddGeometryColumn ( 'gtfs_stops', 'geom', 2264, 'POINT', 2);
UPDATE gtfs_stops SET geom =
ST_Transform(ST_SetSRID(ST_Makepoint(stop_lon,stop_lat),4326),2264);

SELECT
(SELECT ST_Area(ST_Union(ST_Buffer(gtfs_stops.geom,500)))
FROM gtfs_stops, city
WHERE ST_Intersects(city.geom,gtfs_stops.geom)) / (SELECT SUM(ST_Area(geom)) from city)
as CBS_500;

SELECT
(SELECT ST_Area(ST_Union(ST_Buffer(gtfs_stops.geom,500)))
FROM gtfs_stops, region
```

```
WHERE ST_Intersects(region.geom,gtfs_stops.geom)) / (SELECT SUM(ST_Area(geom)) from
region) as CBS_500;

-- For informal stop usage
-- This will take some time to union the line buffer – in production software, this
buffer would be cached
SELECT
(SELECT ST_Area(ST_Union(ST_Buffer(shapelines.geom,500)))
FROM shapelines, city
WHERE ST_Intersects(city.geom, shapelines.geom)) / (SELECT SUM(ST_Area(geom)) from
city) as CBS_500;

SELECT
(SELECT ST_Area(ST_Union(ST_Buffer(shapelines.geom,500)))
FROM shapelines, region
WHERE ST_Intersects(region.geom, shapelines.geom)) / (SELECT SUM(ST_Area(geom)) from
region) as CBS_500;
```

Ratio of the Transit-Pattern Operating Suburban Lines (STR)

Required Data: GTFS, city boundary

Tables Accessed: Stops, trips, routes, stop times, city

Discussion: This query calculates the ratio of the number of routes with stops outside the urban center of the system to the total number of routes.

SQL:

```
SELECT CAST(COUNT(*) as DECIMAL) / CAST((SELECT COUNT(*) FROM gtfs_routes) AS DECIMAL)
FROM
(SELECT t.route_id FROM gtfs_trips t
JOIN gtfs_stop_times st ON st.trip_id = t.trip_id
WHERE st.stop_id IN (SELECT stop_id FROM gtfs_stops, city WHERE NOT
ST_Contains(city.geom, gtfs_stops.geom))
GROUP BY t.route_id) as suburbanlines;
```

On-Time Performance (for low frequency- service)

Required Data: GTFS, Real-time analysis results

Tables Accessed: Routes, trips, stop times, calendar

Discussion: This indicator is calculated using the real-time analysis results, and measures the average deviation from schedule time per route and mode. Additional metrics (standard deviation, z-scores, etc.) may be desirable once data is available and sample results calculated.

Regularity of Headways (for high-frequency service)

Required Data: GTFS, Real-time analysis results

Tables Accessed: Routes, trips, stop times, calendar, frequencies

Discussion: This indicator is calculated using the real-time analysis results, and measures the average deviation from scheduled frequency per route and mode. Additional metrics (standard deviation, z-scores, etc.) may be desirable once data is available and sample results calculated.

Dwell Time Performance

Required Data: GTFS, Real-time analysis results
Tables Accessed: Routes, trips, stop times, calendar
Discussion: This indicator is calculated using the real-time analysis results, and measures the average deviation from scheduled dwell time per route and mode. Additional metrics (standard deviation, z-scores, etc.) may be desirable once data is available and sample results calculated. Dwell time is a somewhat unreliable measure in the real-time analysis results, and indicator results should be understood in this context.

Travel Time Performance

Required Data: GTFS, Real-time analysis results
Tables Accessed: Routes, trips, stop times, calendar
Discussion: This indicator is calculated using the real-time analysis results, and measures the average deviation from scheduled time between stops per route and mode. Additional metrics (standard deviation, z-scores, etc.) may be desirable once data is available and sample results calculated.

Average Service Frequency (ASF) within city and within a bounded area

Required Data: GTFS
Tables Accessed: Routes, trips, stop times, calendar
Discussion: This indicator is calculated in the same fashion as average weekday / weekend frequency.

System coverage (city and bounded area)

Required Data: Stops, boundary files
Tables Accessed: Stops, region, city
Discussion: The calculation for this query is identical to that of CBS500, with the exception that the buffer distance may be variable.

System accessibility (city and bounded area)

Required Data: Population data, GTFS
Tables Accessed: Stops, population
Discussion: This query is a template for selecting a subset of the population and setting the buffer distance of transit stops. In this case, it is the total population measured in 2012 within 500m of a stop.
SQL:

```
SELECT CAST(subpop.subpopsum AS DECIMAL) / CAST(totalpop.totalpopsum AS DECIMAL) as
ratio
FROM (SELECT SUM(pop2012) as subpopsum FROM pop,(SELECT
ST_Union(ST_Buffer(gtfs_stops.geom,500)) as geom FROM gtfs_stops) as coverage WHERE
ST_Intersects(coverage.geom, pop.geom)) AS subpop,
(SELECT SUM(pop2012) as totalpopsum FROM pop) as totalpop;
```

System accessibility for low-income residents (city and bounded area)

Required Data: Population data, GTFS

Tables Accessed: Stops, population

Discussion: Similar to the above indicator, this query is a template to compare a subset of a population to overall population served by the transit system. Configurable items include the subset of the population to choose, as well as the buffer distance from stops.

SQL:

```
SELECT CAST(subpop.subpopsum AS DECIMAL) / CAST(totalpop.totalpopsum AS DECIMAL) as
ratio
FROM (SELECT SUM(low_income) as subpopsum FROM pop,(SELECT
ST_Union(ST_Buffer(gtfs_stops.geom,500)) as geom FROM gtfs_stops) as coverage WHERE
ST_Intersects(coverage.geom, pop.geom)) AS subpop,
(SELECT SUM(pop2012) as totalpopsum FROM pop,(SELECT
ST_Union(ST_Buffer(gtfs_stops.geom,500

)) as geom FROM gtfs_stops) as coverage WHERE ST_Intersects(coverage.geom, pop.geom))
as totalpop;
```

Service frequency, weighted by served population (city and bounded area)

Required Data: GTFS, population

Tables Accessed: Routes, trips, stop times, population

Discussion: This query weights the frequency of each stop, per route, with the population served (within 500m). The frequency is counted for each individual served, and the average is calculated by dividing by the sum of population.

SQL:

```
select sum(freq_wt_pop)/sum(stoppop) as pop_weighted_freq_per_hr from
(select (stop_freq.freq*pop_served_stop.stoppop) as freq_wt_pop,
pop_served_stop.stoppop, stop_freq.stop_id from
  (select (stop_durtn_counts.num/stop_durtn_counts.duration_hrs) as freq,
stop_durtn_counts.stop_id from
       (select count(*) as num, (max(stop_trip_times.arrtime)-
min(stop_trip_times.arrtime)) as duration_hrs, stop_id from
       --Gives all arrivals for one stop without regard to day.
       (select distinct gtfs_stop_times.trip_id, arrival_time, departure_time, stop_id,
```

```
        CAST(substring(arrival_time from 1 for 2) as decimal) +
        cast(substring(arrival_time from 4 for 2) as decimal)/60 as arrtime
        from gtfs_stop_times
        --Gives the trip_id to join the next query on
        join(
        select trip_id, service_id from gtfs_trips
        ) as trips on trips.trip_id=gtfs_stop_times.trip_id
        --Gives all trips which occur on the specified days.
        join(
        select service_id, monday, tuesday, wednesday, thursday, friday
        from gtfs_calendar
        where monday='1' or tuesday='1' or wednesday='1' or thursday='1' or friday='1'
        ) as weekdays
        on weekdays.service_id=trips.service_id
        order by arrtime
        ) as stop_trip_times group by stop_id
        ) as stop_durtn_counts where stop_durtn_counts.duration_hrs > 0.05
  ) as stop_freq

JOIN
  (SELECT gtfs_stops.stop_id, SUM(pop.pop2012) as stoppop
        FROM gtfs_stops
        INNER JOIN pop
        ON ST_Intersects(ST_Buffer(gtfs_stops.geom,500), pop.geom)
        GROUP BY gtfs_stops.stop_id
  ) as pop_served_stop

ON pop_served_stop.stop_id=stop_freq.stop_id
) as weight_pop;
```

Job accessibility

Required Data: GTFS, population data, job data

Tables Accessed:

Discussion: This indicator requires network traversal to be calculated accurately, and cannot be calculated simply in SQL. The approach to calculating this indicator is to set up a series of job locations as destination / origins, calculate the travel time from each job origin, and then select and sum the population that falls into the composite travel shed. Since the technology architecture to calculate this indicator is as of yet undetermined, a full example is temporarily deferred.

Access Index

Required Data:

Tables Accessed:

Discussion: This indicator is currently undefined.
SQL:

Boston Example Results

Number of modes and types

description	numlines
Bus	197
Intercity Rail	12
Underground Rail	8
Ferry	3
Street Level Rail	11

Transit System Length in kilometers

description	modelength
Bus	10036.378380403079088000000
Intercity Rail	1534.6089044789766000
Underground Rail	194.6697746270275200
Street Level Rail	211.4597392782880600

Number of Stops per Route

route_id	num_stops
131	49
505	28
CR-Fitchburg	18
94	40
275	2
79	21
351	24
07	19
CR-Greenbush	10
38	43
222	61
240	82
34	45
880_	20
68	15

436	61
439	50
43	20
221	39
47	39
32	41
9702	22
882_	16
75	27
217	85

Number of stops per mode

description	count
Bus	7696
Intercity Rail	137
Underground Rail	102
Ferry	6
Street Level Rail	150

Weekly number of hours in service - these results are for Monday only

By route:

route_id	duration
70A	15
751	19
50	14
CR-Middleborough	18
326	13
708	14
503	13
112	14
111	21
238	18
434	12
Logan-33	11
4050	1
Boat-F1	17
80	20

By mode:

description	duration

Bus	22
Intercity Rail	21
Underground Rail	22
Ferry	17
Street Level Rail	23

Number of stops per kilometer

route_id	coverage
01	3.7502145377915875
04	3.7175188317261752
05	5.2922152564834317
07	4.3793751226126449
08	5.6597016491916552
09	5.3970313635213175
10	6.1959845609160915
100	4.3392329903062473
101	6.2014935175235907
104	4.1668256815217900
105	4.2772058399081693
106	5.6742376700781739
108	5.9761372413109610
109	5.5054643279267203
11	5.1108343091188745
110	4.8847664659694873
111	3.9747968793396159

Number of stops per kilometer (by mode)

description	coverage
Street Level Rail	1.0262000735488507
Underground Rail	0.55992256737768206987
Intercity Rail	0.09969966911663780589
Bus	0.73811485769256951194

Average distance between stops (by route)

route_id	coverage
01	0.26665141151867976071
04	0.26899661985993617647
05	0.18895678870486448276

```
 07               | 0.22834307909285026316
 08               | 0.17668775882255642157
 09               | 0.18528704627492575333
 10               | 0.16139485019183901042
 100              | 0.23045547501919772105
 101              | 0.16125147872432585714
 104              | 0.23999084109388140625
 105              | 0.23379749243526465116
 106              | 0.17623512763190672245
 108              | 0.16733216785708121489
```

Average distance between stops (by mode)

```
    description    |        coverage
-------------------+------------------------
 Street Level Rail | 0.97446884460040580645
 Underground Rail  |     1.7859612351103442
 Intercity Rail    |    10.0301235586861216
 Bus               | 1.35480269713864458531
```

Average time between consecutive stops on a trip (in minutes)

```
    description    |      avg_min_between_stops
-------------------+------------------------
 Bus               | 0.98250231267564764220
 Intercity Rail    | 5.78788857109534004145
 Underground Rail  | 2.00569748113414382000
 Ferry             | 9.01971726190476190483
 Street Level Rail | 1.55378384223877357018
```

Average system weekday / peak / weekend frequency

```
      buses_per_hr       |     route_id
-------------------------+------------------
 27.725095008971476238 42 | 39
 12.992943548387096774 17 | 171
  9.872173233978898084 66 | 428
 48.000000000000000000 38 | 9701
  2.267971354548725927 16 | CR-Lowell
  9.760096056949339085 43 | 116
 11.544505071418798467 06 | 70
  5.035891830178872357 15 | 236
  1.107359327074822927 45 | 913_
 13.508189718791602639 60 | 45
  1.591041817297887176 46 | CR-Worcester
```

```
 10.36648362816475108598 | 117
  1.43611786269894734362 | CR-Fitchburg
  7.70438552112994282449 | 100
 25.33451839914779233583 | 77
  3.64219653389170785362 | 439
  9.10478444081802356020 | 41
  1.25825246947865956589 | Boat-F3
  7.44912312243229237423 | 94
  1.33467710707008739501 | CR-Franklin
  8.53178877806942060774 | 27
 19.75025675839450091293 | 22
  6.55024726385309147265 | 18
 13.84694918990679130721 | 43
  5.39294337150375765559 | 436
```

System Affordability as a Proportion of Poverty Line Income
10.97%

Ratio of Transit Length to Road Length
0.20958106432611699287

Transit Line Network Density
Region: 0.0112457816953249
City: 5.41088829044538

Coverage of Transit Stops for 500 meters Radius (CBS_500)
City: 1.06696659280942
Region: 0.00352030475304973

Ratio of the Transit-Pattern Operating Suburban Lines (STR)
0.68398268398268398268

Percent of working-age population living within .5 km of a transit stop
0.28810685020257480195

Percent of sub-population within .5 km of a transit stop
0.23048821881903212983

Average Service Frequency, weighted by population
17.035963673858691950 58

Chicago Example Results

Number of modes and types

description	numlines
Bus	126
Underground Rail	8

Transit System Length in kilometers

description	modelength
Bus	9162.03636687290311700000
Underground Rail	620.2784371682611500

route_id | route_type | count

route_id | num_stops

48	57
2	45
X98	2
135	30
94	89
79	99
63W	35
4	106
Y	6
Pink	37
8A	60
205	51
82	99
34	57
68	38
157	56
43	32
5	107

Number of stops per mode

description	count

```
------------------+-------
 Bus              | 11298
 Underground Rail |   296
```

Weekly number of hours in service - these results are for Monday only

By route:

```
route_id | duration
----------+----------
 50       |       20
 112      |       19
 111      |       19
 12       |       24
 J14      |       21
 52A      |       20
 125      |       13
 80       |       24
 206      |       13
 85A      |       16
 9        |       25
 18       |       15
 51       |       16
 146      |       19
 53A      |       21
 86       |       17
 3        |       24
 91       |       19
 155      |       24
 Brn      |       25
 192      |       13
 35       |       21
 67       |       24
 63       |       24
```

By mode:

```
   description    | duration
------------------+----------
 Bus              |       24
 Underground Rail |       24
```

Number of stops per kilometer

```
route_id |        coverage
----------+------------------------
 1        |     4.9938786919840467
```

10	1.5646665884871703
100	5.9471060023054493
103	7.3265884031246463
106	4.4146123354051465
108	5.9045979835421426
11	4.9164065352113857
111	4.8793578310134675
111A	4.6760616956989990
112	6.9034594878388405
115	5.0111795980952442
119	5.2338399167791543
12	7.4172671444257571
120	3.3765359167102014
121	2.5882109326739186
124	4.4005484171337238
125	4.4145275520274712
126	7.7109473663606479
132	2.0711225424436801
134	2.0889753903850233
135	2.5033015794028763
136	2.4077353515506738
143	2.9510555774170951
146	4.1103437617742000

Number of stops per kilometer (by mode)

description	coverage
Underground Rail	0.40626916058931238300
Bus	0.80255084192703922177

Average distance between stops (by route)

route_id	distance
48	0.619872514619883
2	1.22084107744108
X98	25.948
135	1.32462629107981
94	0.668003554230685
79	0.508957523208976
63W	0.667007322929168
4	0.614729313393039

```
 Y         |  8.67699999999999
 Pink      |  2.71516216216215
 8A        | 0.602781193693695
 205       |   0.8031686637618
 82        | 0.605614202222893
 34        | 0.581336370457717
 68        | 0.634233760075867
 157       | 0.623228416149068
 43        | 0.626381913716813
 5         | 0.596820427236315
 156       | 0.588036610008319
 47        | 0.563274876499646
 75        | 0.591964285714286
 76        | 0.589265438165443
 70        | 0.612624919406834
 103       | 0.544735833998403
```

Average distance between stops (by mode)

```
   description    |     distance
------------------+------------------
 Underground Rail | 3.09614382997897
 Bus              | 0.69755568756576
```

Average time between consecutive stops on a trip (in minutes)

By route:

```
 route_id |  avg_min_between_stops
----------+------------------------
 48       |  0.63584763254938693535
 2        |  1.32155197248780136481
 X98      | 12.99999999999999999990
 135      |  1.62249841788452764655
 94       |  0.66022541386438415069
 79       |  0.72102027260736935337
 63W      |  0.55514400167251353856
 4        |  0.70323066165563254681
 Y        |  4.00000000000000000000
 Pink     |  1.66963871087582427788
 8A       |  0.56853512978375241792
 205      |  0.83544419208483461187
 82       |  0.72980171977688700760
 34       |  0.56334214210680942328
 68       |  0.51031616279294297870
```

```
 157      |  1.00040082728926043118
 43       |  0.71312714639309085449
 5        |  0.44626168224299065421
 156      |  0.97125599952462106789
 47       |  0.68226065370483834319
 75       |  0.62483577782085244772
 76       |  0.77085729252957964941
 70       |  0.72584190474715369023
 103      |  0.48644756412739546500


By mode:
   description    | avg_min_between_stops
------------------+-----------------------
 Bus              | 0.75333144128802291409
 Underground Rail | 2.04041373993074064064

Average system weekday frequency (buses per hour)
By route:
      buses_per_hr       | route_id
-------------------------+----------
  8.88700040769585157355 | 85
  9.08481596592020447795 | 75
  3.73048660490879002847 | 100
 11.23325320312689201950 | 55
  5.16289197525055587360 | 108
  4.95406680275577568474 | 95W
  8.73706888544796794151 | 157
  8.72996545277827868018 | 121
  7.20820497658871123797 | 86
  9.76016390288920069430 | 53A
 19.69030486304387848275 | 134
 11.09534473168975347087 | 22
  7.97929770076150466602 | 115
  8.08144589603739069920 | 111
  6.26287396793551092437 | 30
  6.95568847243399742586 | 28
  5.04109589041095890411 | Y

By mode:
          avg           |   description
------------------------+------------------
 8.640730912038599646 76 | Bus
```

```
 6.04372039720269206074 | Underground Rail
```

System Affordability as a Proportion of Poverty Line Income
9.87%

Ratio of Transit Length to Road Length
0.76056620109906795973 -- This is calculated with the city boundary, not regional boundary.

Transit Line Network Density
City: 0.35074626865671641791

Coverage of Transit Stops for 500 meters Radius (CBS_500)
City: 0.905869514700218
-- Regional data not available because it is operated by different agencies (Pace and Metra)

Ratio of the Transit-Pattern Operating Suburban Lines (STR)
0.35074626865671641791 -- The CTA does not provide much service outside Chicago's borders; this is provided by Pace and Metra.

Percent of working-age population living within .5 km of a transit stop
0.58694025909722974378

Percent of sub-population within .5 km of a transit stop
0.23048821881903212983

Population weight frequency per hour, system-wide:
11.009659754054493649 71

Zhengzhou Example Results

Number of modes and types

```
 description | numlines
-------------+----------
 Bus         |      226
```

Transit System Length in kilometers

```
 description |    modelength
-------------+------------------
 Bus         | 3019.73899133209
```

Number of stops by route

route_id	num_stops
289	12
48	25
113	26
962	36
B11	28
2	38
B16	29
131	19
254	20
135	17
275	16
B10	16
79	29
268	17
4	18
38	24

Number of stops per mode

description	count
Bus	7031

Weekly number of hours in service - these results are for Monday only

By route:

route_id	duration
50	16
903	17
273	14
112	16
621	13
111	16
12	15
B1A	18
272	14
203	17
58	17
125	11

```
 80       |       17
 B20      |       17
 206      |       16
 628      |       15
 18       |       15


By mode:
 description | duration
-------------+----------
 Bus         |       24


Number of stops per kilometer
 route_id |     coverage
----------+------------------
route_id |     coverage
----------+--------------------
 50       |  1.88251401189407
 903      |  1.75573782228987
 273      |  3.15135786449161
 621      |  1.30832969908417
 112      |  1.60732380586324
 111      |   1.8308808571235
 12       |  1.17311226684394
 B1A      |  1.26292000398817
 272      |  1.94274612879265
 203      |  2.40711262957642
 58       |  1.99264255058246
 125      | 0.860030101053538
 80       |  1.99474716579672
 B20      |  1.70614869742109
 206      |  2.19178082191781
 628      |  1.65584002037957
 18       |  1.49142431021625
Number of stops per kilometer (by mode)

description |     coverage
-------------+------------------
 Bus         | 1.82267408401811
Average distance between stops (by route)

route_id |   routeavgstopdist
```

```
----------+-------------------
 50       | 0.531204545454545
 903      | 0.569561119721041
 273      | 0.317323529411765
 621      |  0.76433333333333
 112      | 0.622152173913043
 111      | 0.546185185185181
 12       |  0.85243333333333
 B1A      | 0.791815789473683
 272      | 0.514735294117645
 203      | 0.415435483870969
 58       | 0.501846153846156
 125      |           1.16275
 80       |  0.50131666666667
 B20      | 0.586115384615385
 206      | 0.456249999999999
```

Average distance between stops (by mode)

```
description |     coverage
------------+-------------------
 Bus        | 0.548644438832139
```

Average time between consecutive stops on a trip (in minutes)

By route:

```
 route_id | avg_min_between_stops
----------+-----------------------
 289      | 4.56439393939393939394
 48       | 4.80000000000000000000
 113      | 4.79986962190352020861
 962      | 4.86111111111111111111
 B11      | 4.82142857142857142857
 2        | 4.86842105263157894737
 B16      | 4.82758620689655172414
 131      | 4.72953216374269005848
 254      | 4.75000000000000000000
 135      | 4.67436974789915966387
 275      | 4.68750000000000000000
 B10      | 4.67708333333333333333
 79       | 4.82450738916256157635
 268      | 4.70588235294117647059
```

```
 4         | 4.71405228758169934641
```

By mode:

```
 description | avg_min_between_stops
-------------+-----------------------
 Bus         | 4.74547519609168491380
```

Average weekday frequency (stops per hour)

```
     buses_per_hour      | route_id
-------------------------+----------
  9.82578397212543554007 | 1
 12.29234199468313180688 | 10
 12.06783493499152063313 | 100
 12.05555555555555555556 | 101
 12.42156108909524654482 | 102
 12.35992372477355792150 | 103
 12.06451612903225806452 | 104
 12.06666666666666666667 | 105
 12.06666666666666666667 | 106
 12.33932391138273491215 | 107
 12.05555555555555555556 | 108
 12.51366828131329712378 | 109
 12.06896551724137931034 | 11
 12.07407407407407407407 | 111
 12.28481240981240981241 | 112
 12.49397877984084880637 | 113
 12.07407407407407407407 | 114
```

By mode:

```
     buses_per_hour      | description
-------------------------+-------------
 12.18434079140008432690 | Bus
```

Additional indicators not calculated due to lack of road network and population data